Problem Statement : Develop a script to automatically sort files into folders based on their extensions, names, or creation dates.

1. **File Sorter** — Automatically sorts files based on extension, name pattern, or creation date.
2. **Calculator** — Performs basic arithmetic operations with error handling, optional GUI, and operation history logging using SQLite.

Code :

```
#file_sorter.py

import os

import shutil

from datetime import datetime


def sort_files_by_extension(directory):
    for file in os.listdir(directory):
        if os.path.isfile(os.path.join(directory, file)):
            ext = file.split('.')[-1]
            ext_folder = os.path.join(directory, ext)
            os.makedirs(ext_folder, exist_ok=True)
            shutil.move(os.path.join(directory, file), os.path.join(ext_folder, file))
    print("Files sorted by extension.")


def sort_files_by_date(directory):
    for file in os.listdir(directory):
        file_path = os.path.join(directory, file)
        if os.path.isfile(file_path):
            creation_time = os.path.getctime(file_path)
            date_folder = datetime.fromtimestamp(creation_time).strftime('%Y-%m-%d')
            folder_path = os.path.join(directory, date_folder)
            os.makedirs(folder_path, exist_ok=True)
            shutil.move(file_path, os.path.join(folder_path, file))
```

```
    print("Files sorted by date.")

def sort_files_by_name(directory, keyword):
    name_folder = os.path.join(directory, keyword)
    os.makedirs(name_folder, exist_ok=True)
    for file in os.listdir(directory):
        if keyword in file:
            shutil.move(os.path.join(directory, file), os.path.join(name_folder, file))
    print(f"Files with '{keyword}' moved.")

if __name__ == "__main__":
    path = input("Enter the path to the directory: ").strip('"').strip("'")
    print("1. Sort by extension\n2. Sort by creation date\n3. Sort by name keyword")
    choice = input("Enter choice (1/2/3): ")

    if choice == '1':
        sort_files_by_extension(path)
    elif choice == '2':
        sort_files_by_date(path)
    elif choice == '3':
        keyword = input("Enter keyword to search in filenames: ")
        sort_files_by_name(path, keyword)
    else:
        print("Invalid choice.")

#calculator.py
import os
import logging
from db_helper import log_operation
```

```
# Ensure 'logs' directory exists before logging
log_folder = 'logs'
if not os.path.exists(log_folder):
    os.makedirs(log_folder)

#  Set up logging
log_file = os.path.join(log_folder, 'operation_log.txt')
logging.basicConfig(filename=log_file, level=logging.INFO)

def calculate():
    while True:
        try:
            expr = input(">> ")
            if expr.lower() == 'exit':
                break
            result = eval(expr)
            print("Result:", result)
            log_operation(expr, result)
            logging.info(f"{expr} = {result}")
        except ZeroDivisionError:
            print("Error: Division by zero.")
        except Exception as e:
            print("Invalid input:", e)

if __name__ == "__main__":
    calculate()

#calculator_gui.py
import tkinter as tk
from db_helper import log_operation
```

```
def calculate():
    try:
        expr = entry.get()
        result = str(eval(expr))
        label_result.config(text="Result: " + result)
        log_operation(expr, result)
    except ZeroDivisionError:
        label_result.config(text="Error: Division by zero")
    except Exception:
        label_result.config(text="Invalid Input")

root = tk.Tk()
root.title("Calculator")

entry = tk.Entry(root, width=30)
entry.pack(pady=10)

tk.Button(root, text="Calculate", command=calculate).pack()
label_result = tk.Label(root, text="Result: ")
label_result.pack()

root.mainloop()

#db_helper.py
import sqlite3

def init_db():
    conn = sqlite3.connect("history.db")
    cursor = conn.cursor()
```

```
    cursor.execute('''CREATE TABLE IF NOT EXISTS operations (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    expression TEXT,
                    result TEXT,
                    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP)''')
    conn.commit()
    conn.close()

def log_operation(expr, result):
    init_db()
    conn = sqlite3.connect("history.db")
    cursor = conn.cursor()
    cursor.execute("INSERT INTO operations (expression, result) VALUES (?, ?)", (expr, result))
    conn.commit()
    conn.close()
```

Test Cases:

calculator.py:

- Input: 3 + 5 * 2 → Output: 13
- Input: 8 / 0 → Output: Error: Division by zero

file_sorter.py:

- Sorts .txt, .jpg, etc. into respective folders.
- Groups files like DSA_2023.txt, DSA_2024.txt into report/

Technology Used:

- Python
- Tkinter (GUI)
-SQLite (operation history)
-Logging module